ELSEVIER

Theoretical Computer Science 165 (1996) 463–474

Theoretical Computer Science

Note

Nonreturning PC grammar systems can be simulated by returning systems[1]

Sorina Dumitrescu

University of Bucharest, Faculty of Mathematics, Str. Academiei 14, 70109 Bucureşti, Romania

Received September 1995; revised November 1995
Communicated by A. Salomaa

Abstract

One proves that the generative capacity of nonreturning parallel communicating (PC) grammar systems with context-free rules, centralized or not, does not overpass that of the noncentralized returning PC grammar systems. This strengthens previous results in this area and clarifies the returning–nonreturning relationship.

1. Introduction

The *parallel communicating (PC) grammar systems* have been introduced in [9] as a grammatical model of parallel computing. Motivations and bibliographical details can be found in [2].

In short, such a system consists of several grammars that work synchronously, on their own sentential forms, and communicate by request. More specifically, in the nonterminal alphabet of the system there are special symbols Q_j, called *query* symbols, associated in a one to one manner to the components of the system. When a component G_i introduces a query symbol Q_j, the component G_j has to send its current string (providing it does not contain query symbols) to the grammar G_i for replacing all the occurrences of Q_j. One component is distinguished as the *master* and the language generated by it, alone or involving communications, is the language generated by the system.

There are two important classifications of PC grammar systems, with respect to the *communication graph* and to the *returning feature*: a system is called *centralized* if only the master is allowed to introduce query symbols, and *noncentralized*, in the nonrestricted case; if any component, after communicating, resumes working from its

[1] Research supported by the Academy of Finland, Project 11281.

SSDI 0304-3975(95)00258-8

axiom, then the system is called *returning* and it is *nonreturning* if after communication each component continues to process its current string.

This way, four basic classes of PC grammar systems are obtained: *centralized returning*, *centralized nonreturning*, *noncentralized returning*, *noncentralized nonreturning*. Clearly, each centralized system is by definition also noncentralized, but the relationship between returning and nonreturning systems is much more complex. For instance, the families of languages generated by centralized returning and centralized nonreturning systems with right-linear rules are incomparable [2, 3]. There is no difference between returning and nonreturning context-sensitive systems: in the centralized case both types of systems characterize the context-sensitive languages, in the noncentralized case they characterize the recursively enumerable languages, see [2, 4, 10].

For a while no relation has been known for PC grammar systems with context-free components, between the families generated in the returning and the nonreturning manner. Recently, it has been proved in [7], that a centralized nonreturning PC grammar system with context-free rules can be simulated by a noncentralized returning system.

One strengthens here this result, by proving that the nonreturning feature can be simulated by the returning one even when starting from noncentralized PC grammar systems. Thus, the centralization is not a condition for obtaining such a result. Moreover, the proof gives, as particular cases, similar results for right-linear and linear PCGS's. (For linear PC grammar systems the assertion is proved also in [12].) The use of noncentralization is essential in this proof, and we *conjecture* that the corresponding inclusion does not hold in the centralized case: one cannot simulate a centralized nonreturning PCGS with context-free rules by a centralized returning system.

The result presented here is significant for the theory of PC grammar systems for at least two reasons: (1) it settles a relation between two basic families; (2) many papers about PC grammar systems are devoted to returning systems; this is, for instance, the case of all complexity researches in this area [1, 5, 6, 8, etc.]. One sees now that the nonreturning systems are, in fact, unimportant from the generative point of view, they are covered by the returning systems. This, however, does not imply that the complexity of the two types of systems is necessarily the same: the construction here increases the number of components of the involved systems from n to a polynomial of degree two and, moreover, to each step in the initial system about $2n$ steps correspond in the new one when generating a string.

As a basic idea of this proof one extends the idea used in [7]: at the same moment a communication from a component G_i to a component G_j is performed, the string of G_i is also communicated to a wittness component G_i'. Thus a copy of the current string of G_i is provided, such that, by a communication from G_i' to G_i, one simulates in the returning way a nonreturning communication from G_i to G_j. The construction is much more complex when one starts from a noncentralized system because of multiple queries and, mainly, to linked queries, from one component to another one. Discussions about these difficulties and examples illustrating them can be found in the next section.

2. Parallel communicating grammar systems

One uses the following notations. For a finite alphabet V, V^* is the free monoid generated by V under the operation of concatenation. Its elements are called strings or words; λ is the empty string, $V^+ = V^* - \{\lambda\}$. For $x \in V^*$, $|x|$ is the length of x and if $U \subseteq V$, $|x|_U$ is the number of all occurrences of symbols of U in x. For more details in formal language theory the reader is referred to [11].

Definition. A *PC grammar system* of degree n, $n \geqslant 1$, is a construct

$$\gamma = (N, K, T, G_1, \ldots, G_n),$$

where N, K, T are pairwise disjoint alphabets, with $K = \{Q_1, \ldots, Q_n\}$, $G_i = (N \cup K, T, P_i, S_i)$, $1 \leqslant k \leqslant n$; the elements of N are *nonterminal* symbols, those of T are *terminals*; the elements of K are called *query symbols*; the grammars G_i are the *components* of the system. The query symbols are associated in a one to one manner to the components, by their indices. When discussing the type of the components in Chomsky hierarchy, the query symbols are interpreted as nonterminals.

For $(x_1, \ldots, x_n), (y_1, \ldots, y_n)$, with $x_i, y_i \in (N \cup T \cup K)^*$, $1 \leqslant i \leqslant n$ (we call such an n-tuple a *configuration*), $x_1 \notin T^*$, we write $(x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)$ if one of the following two cases holds:

(i) $|x_i|_K = 0$ for all $1 \leqslant i \leqslant n$; then $x_i \Rightarrow_{P_i} y_i$ or $x_i = y_i \in T^*$, $1 \leqslant i \leqslant n$;

(ii) there is i, $1 \leqslant i \leqslant n$, such that $|x_i|_K > 0$; we write such a string x_i as

$$x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1},$$

for $t \geqslant 1$, $z_i \in (N \cup T)^*$, $1 \leqslant i \leqslant t+1$; if $|x_{i_j}|_K = 0$ for all $l \leqslant j \leqslant t$, then

$$y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1},$$

[and $y_{i_j} = S_{i_j}, 1 \leqslant j \leqslant t$]; otherwise $y_i = x_i$. For all unspecified i we have $y_i = x_i$.

Point (i) defines a *rewriting* step (componentwise, on all components whose current strings are not terminal), (ii) defines a *communication* step: the query symbols Q_{i_j} introduced in some x_i are replaced by the associated strings x_{i_j}, providing that these strings do not contain further query symbols. The communication has priority over rewriting. The work of the system is blocked when circular queries appear, as well as when no query symbol is present but point (i) is not performed because a component cannot rewrite its sentential form, although it is a nonterminal string.

The above considered definition of the relation $\Rightarrow$ is given for the *returning* PC grammar systems: after communicating, a component resumes working from its axiom. If the brackets, [and $y_{i_j} = S_{i_j}$, $1 \leqslant i \leqslant t$], are removed, then we obtain the definition of a derivation step in the *nonreturning* PCGS's: after communicating, a component continues to process the current string.

The language generated by γ is the language generated by its first component, called the *master* grammar (G_1 above), when starting from $(S_1, \ldots, S_n)$, that is

$$L(\gamma) = \{w \in T^* \mid (S_1, \ldots, S_n) \Rightarrow^* (w, \alpha_2, \ldots, \alpha_n), \text{ for } \alpha_i \in (N \cup T \cup K)^*, 2 \leqslant i \leqslant n\}.$$

(One does not care about the strings in the components $2,\ldots,n$ in the last configuration of a terminal derivation; moreover, the work of γ stops when a terminal string is obtained in the first component since no more derivation steps can be done.)

The systems in which only the master may introduce query symbols are called *centralized* and the other ones are called *noncentralized.* Denote by $PC(X)/CPC(X)$ the family of languages generated by the returning noncentralized/centralized PC grammar systems with rules of type X (and of arbitrarily degree) and by $NPC(X)/NCPC(X)$ the family of languages generated by nonreturning noncentralized/centralized PC grammar systems with rules of type X. One considers here $X = RL$ (right-linear) or $X = CF$ (context-free).

3. Simulating nonreturning systems by returning systems

The difference between the returning and the nonreturning systems consists of the way the communication steps are performed. We give here an example for illustrating this fact. Take first the centralized case. Suppose that the PC grammar system has three components and that after a rewriting step the following configuration is obtained.

$$(u_1Q_2u_2Q_3u_3,\ x_2,\ x_3),$$

where u_1, u_2, u_3, x_2, x_3 do not contain query symbols. If the system is nonreturning, then after the communication step one has

$$(u_1x_2u_2x_3u_3,\ x_2,\ x_3).$$

If the system is returning, the configuration obtained after communication is

$$(u_1x_2u_2x_3u_3,\ S_2,\ S_3).$$

Note that in both cases the same strings are communicated to the master (G_1), the difference being that in the returning system no copy of the transmitted strings remains in the originate grammars.

Take now a noncentralized system with three components. Then it is possible to obtain configurations in which two components introduce query symbols. Consider such a configuration

$$(u_1Q_2u_2Q_3u_3,\ v_1Q_3v_2,\ x_3), \qquad (\alpha)$$

where $u_1, u_2, u_3, v_1, v_2, x_3$ do not contain query symbols. For satisfying all the query symbols, two communication steps are needed. In the nonreturning case these steps are performed as follows

$$\begin{aligned}(\alpha) &\Rightarrow (u_1Q_2u_2Q_3u_3,\ v_1x_3v_2,\ x_3)\\ &\Rightarrow (u_1v_1x_3v_2u_2x_3u_3,\ v_1x_3v_2,\ x_3)\end{aligned}$$

and in the returning case

$$(\alpha) \Rightarrow (u_1Q_2u_2Q_3u_3,\ v_1x_3v_2,\ S_3)$$

$$\Rightarrow (u_1v_1x_3v_2u_2S_3u_3,\ S_2,\ S_3)$$

Note that here the difference is two sided: the first one is the same as in the centralized case, while the second one consists of the fact that the strings communicated to the grammar G_1 are not the same in the two considered situations.

This example reveals the increased complexity of simulating nonreturning noncentralized PC grammar systems by returning PC grammar systems, compared to the centralized case.

Theorem. $NPC(X) \subseteq PC(X)$, $X \in \{RL, CF\}$.

Proof. Consider first the context-free case. Let γ be a noncentralized nonreturning PC grammar system

$$\gamma = (N, K, T, G_1, \ldots, G_n),$$

for some $n \geqslant 2$ (the case $n = 1$ is trivial).

We construct a noncentralized returning PC grammar system γ'

$$\gamma' = (N', K', T, G_{11}, G_{11}^{(2)}, \ldots, G_{11}^{(2n-2)}, G_{12}, G_{12}^{(2)}, \ldots, G_{12}^{(2n-2)}, G_{13}, \ldots,$$
$$G_{n1}, G_{n1}^{(2)}, \ldots, G_{n1}^{(2n-2)}, G_{n2}, G_{n2}^{(2)}, \ldots, G_{n2}^{(2n-2)}, G_{n3}, G_a),$$

where $N' = N \cup \{S_{kj} \mid 1 \leqslant k \leqslant n,\ 1 \leqslant j \leqslant 3\} \cup \{S, Z, Z'\}$. The query symbols are denoted using the same symbols as the associated grammars, also the production set and the axiom of each grammar, excepting the query symbols for G_{kj} which are $Q_{kj}^{(1)}$, respectively, and the axioms of G_a and of $G_{kj}^{(s)}$, which are S, for all $1 \leqslant k \leqslant n,\ 1 \leqslant j \leqslant 2,\ 2 \leqslant s \leqslant 2n-2$. The production sets of the components of γ' are

$$P_{k1} = \{S_{k1} \to Q_{k2}^{(2n-2)}, S_{k1} \to Q_{k3}\} \cup P'_{k1},$$

$$P_{k2} = \{S_{k2} \to Q_{k1}^{(2n-2)}, S_{k2} \to Z'\} \cup P'_{k2},$$

$$P'_{kj} = \{A \to x \mid A \to x \in P_k,\ x \in (N \cup T)^*,\ A \in N\}$$

$$\cup \{A \to u_1 Q_{i_1 j}^{(s_1)} u_2 Q_{i_2 j}^{(s_2)} \cdots u_t Q_{i_t j}^{(s_t)} u_{t+1} \mid A \to u_1 Q_{i_1} u_2 Q_{i_2} \cdots$$

$$u_t Q_{i_t} u_{t+1} \in P_k,\ t \geqslant 1,\ u_1, \ldots, u_{t+1} \in (N \cup T)^*,\ A \in N,$$

$$1 \leqslant i_1, \ldots i_t \leqslant n,\ 2 \leqslant s_1, \ldots s_t \leqslant 2n-2,\ s_1, \ldots, s_2 \text{ even numbers}\},$$

$$P_{kj}^{(s)} = \{S \to Z,\ Z \to Q_{kj}^{(s-1)}\}, \quad 2 \leqslant s \leqslant 2n-2,$$

$$P_{k3} = \{S_{k3} \to S_k,\ S_k \to Z',\ Z' \to Z'\},$$

$$P_a = \{S \to Q_{12}Q_{12}^{(2)} \cdots Q_{12}^{(2n-2)} \cdots Q_{n2}Q_{n2}^{(2)} \cdots Q_{n2}^{(2n-2)},\ Z \to Z\}.$$

for all k, $1 \leqslant k \leqslant n$, and $j = 1, 2$.

The idea of this construction is to consider for each component G_k, $1 \leqslant k \leqslant n$, of the initial system, a couple of grammars G_{k1} and G_{k2}, which simulate the functioning of G_k. They work, in fact, on the same sentential form, which, at one rewriting step is in one of them, at the next rewriting step is in the other and so on. Hence, after each rewriting step in the new system γ', a sequence of consecutive communication steps follows, after that the sentential form passes from one of the grammars G_{k1}, G_{k2}, to the other. One has also introduced the "assistant" grammars $G_{kj}^{(s)}$, $j = 1, 2$, $2 \leqslant s \leqslant 2n-2$, that help to perform this transmission of the sentential form which circulates between G_{k1} and G_{k2}. More precisely, after the sentential form is rewritten in G_{k1} (one step), a sequence of communications follows, during which, after all (possible) occurrences of query symbols in the sentential form are satisfied, it is transmitted first to the component $G_{k1}^{(2)}$, then (from here) to $G_{k1}^{(3)}$ and so on, until it arrives in $G_{k1}^{(2n-2)}$, from where it is sent to G_{k2}. During these communications the grammars $G_{k2}^{(s)}$, $2 \leqslant s \leqslant 2n-2$, are inactive. After this sequence of communications is finished, the sentential form is rewritten one step in G_{k2}, and then another sequence of communications follows. After all occurrences of query symbols in the sentential form are satisfied (if any), the string is sent, this time, to $G_{k2}^{(2)}$, from here to $G_{k2}^{(3)}$, etc. until it arrives to $G_{k2}^{(2n-2)}$, from where it leaves to G_{k1}.

The grammar system γ is nonreturning. This means that each component, after communicating the current string, keeps a copy of it and continues to derive it. Not the same is happening in γ' which is a returning PC grammar system. But in γ', by passing the sentential form from one of the components G_{k1}, G_{k2} to the other after each rewriting step, we ensure that the string is saved after a communication to another grammar G_{mj}. In the same time γ is noncentralized. Hence at one moment it is possible that query symbols appear in several components and more consecutive communication steps might be necessary for satisfying all of them. Their number is at most $n-1$ (at least one component must have a string without query symbols; otherwise the derivation is blocked). It also may happen that one grammar communicates its string at each of these steps. That is why one had to introduce in γ' the assistant grammars, for saving the sentential form after a larger number of consecutive communication steps.

The production rules of the grammars G_{k1} and G_{k2}, other than those which rewrite the axiom, are those of G_k excepting the rules that introduce query symbols. For each such rule r of G_k, there have been introduced in G_{kj}, $j = 1, 2$, all the rules obtained by modifying the query symbols, such that, if in the initial rule r a symbol refers to the grammar G_m, then, in the modified rule it refers to any of the assistant grammars $G_{mj}^{(s)}$, $2 \leqslant s \leqslant 2n-2$, where s is an even number. At each use of r, not any of the modified rules is the right one. It depends on the other components of γ, too, which is

the rule (or the rules) that must be applied in G_{kj}, for simulating by γ' the behaviour of γ at the communication steps that follow. But, if a wrong rule is used, then either in the sentential form of G_{kj}, or in that of another grammar G_{mj}, the symbol S (the axiom of the assistant grammars) will appear and this cannot be rewritten in any of the grammars G_{ij}, $1 \leqslant i \leqslant n$, $j = 1,2$. Hence, the respective sentential form will not participate anymore at the generation of a terminal word. This way one ensures that parasitic words are not generated.

Since the component G_{kj} cannot receive directly the current string of another component G_{mj}, but only after this has arrived at least in $G_{mj}^{(2)}$ (hence, after at least two communication steps), the number of the communication steps after which all the query symbols in G_{kj} can be satisfied increases (of at least two times) compared to that of the corresponding sequence of communications in the initial system. That is why one had to add $2n-2$ assistant grammars for each G_{kj}, $j = 1,2$, $1 \leqslant k \leqslant n$.

The role of each component G_{k3} of γ' is to introduce after the first step of the derivation, the symbol S_k and to send it to G_{k1}. This way it can start in G_{k1} the simulation of the work of G_k. If farther the grammar G_{k1} will ask again the string in G_{k3}, then after the communication, the derivation will get blocked (the received string Z' cannot be rewritten in G_{k1}). Thus one ensures that G_{k1} asks only from $G_{k2}^{(2n-2)}$ and follows the behaviour described above.

The grammar G_a has the role to introduce one step difference between the functioning of the group $G_{k2}, G_{k2}^{(2)}, \ldots, G_{k2}^{(2n-2)}$ and that of the group $G_{k1}, G_{k1}^{(2)}, \ldots, G_{k1}^{(2n-2)}$, for each k, $1 \leqslant k \leqslant n$. The two groups start the derivation by the same type of rules (excepting G_{k1}), but after the first rewriting step, the components in the first group communicate their strings to G_a and resume working from the beginning. After this communication, G_a cannot influence anymore the functioning of the system.

In the sequel one gives the formal proof of the fact that $L(\gamma') = L(\gamma)$.

($\subseteq$) Let D' be a terminal derivation in γ'. Its initial configuration is

$$\begin{array}{ccccccccccccc} (\ldots, & S_{k1}, & \ldots, & S, & \ldots, & S_{k2}, & \ldots, & S, & \ldots, & S_{k3}, & \ldots, & S), \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{1}$$

where $1 \leqslant k \leqslant n$, $2 \leqslant s \leqslant 2n-2$. After the first rewriting step, in each component $G_{kj}^{(s)}$, $j = 1,2$, $2 \leqslant s \leqslant 2n-2$, $1 \leqslant k \leqslant n$, the symbol Z is obtained, in G_{k3}, the symbol S_k and in G_a the string $Q_{12}Q_{12}^{(2)} \cdots Q_{12}^{(2n-2)} \cdots Q_{n2} \cdots Q_{n2}^{(2n-2)}$, for all $j = 1,2$, $2 \leqslant s \leqslant 2n-2$, $1 \leqslant k \leqslant n$. Depending on the strings obtained in G_{k1} and G_{k2} one has more situations.

Case a: In G_{k1} the symbol Q_{k3} is produced and in G_{k2} the symbol Z', for all k, $1 \leqslant k \leqslant n$. Then

$$\begin{array}{ccccccccccccc} (1) \Rightarrow (\ldots, & Q_{k3}, & \ldots, & Z, & \ldots, & Z', & \ldots, & Z, & \ldots, & S_k, & \ldots, & Q_{12}Q_{12}^{(2)} \cdots Q_{n2}^{(2n-2)}) \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{1'}$$

$$(1') \Rightarrow (\ldots, S_k, \ldots, Z, \ldots, S_{k2}, \ldots, S, \ldots, S_{k3}, \ldots, (Z')^n Z^{n(2n-2)}). \tag{2'}$$

Case b: There is a k_0, $1 \leqslant k_0 \leqslant n$, such that in $G_{k_0 1}$ the symbol $Q_{k_0 2}^{(2n-2)}$ is obtained. Then after the communication step one has the symbol Z in $G_{k_0 1}$ and it cannot be

rewritten in this grammar. Hence, the derivation is blocked (no other grammar asks for the string of G_{k_01}).

Case c: There is a k_0, $1 \leqslant k_0 \leqslant n$, such that in G_{k_01} the symbol Q_{k_03} is produced, and in G_{k_02} the symbol $Q_{k_01}^{(2n-2)}$. Then

$$\begin{array}{lllllllllllll} (1) \Rightarrow (\ldots, & Q_{k_03}, & \ldots, & Z, & \ldots, & Z, & Q_{k_01}^{(2n-2)}, & \ldots, & Z, & \ldots, & S_{k_0}, & \ldots, & Q_{12}\cdots Q_{n2}^{(2n-2)}) \\ & G_{k_01} & & G_{k_01}^{(s)} & & G_{k_01}^{(2n-2)} & G_{k_02} & & G_{k_02}^{(s)} & & G_{k_03} & & G_a \\ \Rightarrow^* (\ldots, & S_{k_0}, & \ldots, & Z, & \ldots, & S, & S_{k_02}, & \ldots, & S, & \ldots, & S_{k_03}, & \ldots, & \alpha), \end{array} \tag{2'}$$

where $\alpha \in \{Z, Z'\}^+$. If at the next rewriting step one applies in G_{k_02} the rule $S_{k_02} \to Z'$, the derivation will be blocked (Z' cannot be rewritten in G_{k_02} and cannot be required by another grammar). Consequently, one applies the rule $S_{k_02} \to Q_{k_01}^{(2n-2)}$. Then

$$\begin{array}{lllllllllll} (2') \Rightarrow (\ldots, & \alpha_{k_0}, & \ldots, & Q_{k_01}^{(s-1)}, & \ldots, & Z, & Q_{k_01}^{(2n-2)} & \ldots, & Z, & \ldots, & S_{k_0}, & \ldots, & \alpha) \\ & G_{k_01} & & G_{k_01}^{(s)} & & G_{k_01}^{(2n-2)} & G_{k_02} & & G_{k_02}^{(s)} & & G_{k_03} & & G_a \\ \Rightarrow^* (& & & \ldots & & & Z & & & \cdots & & &). \end{array}$$

After the communication, in G_{k_02} the symbol Z is present and it cannot be rewritten. Hence, the derivation will be blocked. (Note that no string α_k can have the symbol $Q_{k_02}^{(1)}$.)

In conclusion, only *Case* a holds.

Assume that at one moment one has in D' the following configuration (similar to (2))

$$\begin{array}{lllllllllll} (\ldots, & x_k, & \ldots, & Z, & \ldots, & S_{k2}, & \ldots, & S, & \ldots, & X_k, & \ldots, & \beta), \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{3}$$

where $\beta \in \{Z, Z'\}^+$, $X_k \in \{S_{k3}, S_k, Z'\}$, $1 \leqslant k \leqslant n$, and $(x_1, \ldots, x_n)$ is a configuration in a terminal derivation in γ. Starting from (3), one cannot apply the rule $S_{k2} \to Z'$ in G_{k2} since the derivation will get blocked. Consequently, one has

$$\begin{array}{lllllllllll} (3) \Rightarrow (\ldots, & y_k', & \ldots, & Q_{k1}^{(s-1)}, & \ldots, & Q_{k1}^{(2n-2)}, & \ldots, & Z, & \ldots, & Y_k, & \ldots, & \beta), \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{4}$$

where $Y_k \in \{S_k, Z'\}$, $1 \leqslant k \leqslant n$. This step corresponds to the rewriting step in γ $(x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)$, where y_k is the string obtained from y_k' by replacing each query symbol $Q_{t1}^{(s)}$ by Q_t, for all k, t, s, $1 \leqslant k$, $t \leqslant n$, $2 \leqslant s \leqslant 2n-2$.

In the system γ' a sequence of communication steps follows (we shall analyse it later):

$$\begin{array}{lllllllllll} (4) \Rightarrow (\ldots, & S_{k1}, & \ldots, & S, & \ldots, & z_k', & \ldots, & Z, & \ldots, & Y_k, & \ldots, & \beta). \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{5}$$

Assume now that one has at one moment in D', the configuration

$$\begin{array}{lllllllllll} (\ldots, & S_{k1}, & \ldots, & S, & \ldots, & z_k, & \ldots, & Z, & \ldots, & Y_k, & \ldots, & \beta), \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{6}$$

where Y_k are as above and $(z_1,\dots,z_n)$ is a configuration of a terminal derivation in γ. After the rewriting step which follows, in G_{k3} one obtains Z' and in $G_{k1}^{(s)}$ the symbol Z, for $1\leqslant k\leqslant n$, $2\leqslant s\leqslant 2n-2$. If in a grammar G_{k_01} the symbol Q_{k_03} is produced, then, after the communication step, the derivation will be blocked (Z' cannot be rewritten in G_{k_01} and it is not required by another grammar). For avoiding this one applies the rule $S_{k1}\to Q_{k2}^{(2n-2)}$ in $G_{k1}, 1\leqslant k\leqslant n$. Hence

$$\begin{array}{llllllllllll} (6)\Rightarrow(\dots, & Q_{k2}^{(2n-2)}, & \dots, & Z, & \dots, & w_k', & \dots, & Q_{k2}^{(s-1)}, & \dots, & Z', & \dots, & \beta). \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{7}$$

This rewriting step corresponds to the rewriting step in $\gamma\colon (z_1,\dots,z_n)\Rightarrow(w_1,\dots,w_n)$, where w_k is obtained from w_k' by replacing any symbol $Q_{t2}^{(s)}$ by Q_t, $1\leqslant k,\ t\leqslant n,\ 2\leqslant s\leqslant 2n-2$.

Then a sequence of communication steps follows in γ'.

$$\begin{array}{llllllllllll} (7)\Rightarrow^*(\dots, & v_k', & \dots, & Z, & \dots, & S_{k2}, & \dots, & S, & \dots, & Z', & \dots, & \beta). \\ & G_{k1} & & G_{k1}^{(s)} & & G_{k2} & & G_{k2}^{(s)} & & G_{k3} & & G_a \end{array} \tag{8}$$

In what follows we shall analyse this sequence of communication steps. Denote by w_k'' the word obtained after all the query symbols in w_k' have been satisfied (if there are such symbols, if not, then $w_k''=w_k'$). After w_k'' is obtained in G_{k2}, this is sent, at the next communication step, to the grammar $G_{k2}^{(2)}$, at the next step it leaves $G_{k2}^{(2)}$ to $G_{k2}^{(3)}$ and so on until, after $2n-3$ communication steps it arrives in $G_{k2}^{(2n-2)}$. After the $(2n-2)$-th communication step it reaches G_{k1}. The string which has passed through all the grammars associated to G_{k2}, arrives in G_{k1} and remains here until the sequence of communications finishes. One concludes that $w_k''=v_k', 1\leqslant k\leqslant n$. Note that, after the jth communication step, $1\leqslant j\leqslant 2n-3$, after obtaining v_k' in G_{k2}, the string v_k' is in the component $G_{k2}^{(j+1)}$, in all the components G_{k2}, $G_{k2}^{(s)}$, $2\leqslant s\leqslant j$, the axiom is present and in $G_{k2}^{(s)}$, $j+2\leqslant s\leqslant 2n-2$, query symbols appear, for all k, $1\leqslant k\leqslant n$. This remark is very important for the sequel.

If none of the strings w_k' contains query symbols, then it is clear that $w_k=w_k'$ (where $w_1,\dots,w_n$ have been defined above). As $w_k''=w_k'$, it follows that $v_k'=w_k$, $1\leqslant k\leqslant n$. Hence, $(v_1',\dots,v_n')$ is a configuration in a terminal derivation of γ.

Assertion 1. If there is at least one string w_k', $1\leqslant k\leqslant n$, which contains query symbols, then clearly, w_k also has query symbols. Let $(v_1,\dots,v_n)$ be the configuration obtained in γ after satisfying all the query symbols in the configuration $(w_1,\dots,w_n)$. If, in addition, $v_k'\in(N\cup T)^*$, $1\leqslant k\leqslant n$, (hence, v_k' does not contain nonterminal symbols other than the nonterminals of γ), then $v_k'=v_k$, $1\leqslant k\leqslant n$.

Proof. Let n_1 be the number of the communication steps in the sequence of communications (7) $\Rightarrow^*$ (8). Make the following notations: M_0' is the set of all indices k, $1\leqslant k\leqslant n$, for which w_k' does not contain query symbols; for each i, $1\leqslant i\leqslant n$, M_i' is the set of all indices k, $1\leqslant k\leqslant n$, such that all the query symbols in w_k' are satisfied at the ith communication step (of this sequence).

One proves by induction on i, $0 \leqslant i \leqslant n_1$, that, for all $k \in M'_i$, $v'_k = v_k$ holds. Let $i = 0$. Since $k \in M'_0$, it follows that w'_k does not contain query symbols, hence, w_k either. Then $w'_k = w_k$ (by the definition of w_k), $w''_k = w'_k$, $w''_k = v'_k$ and $v_k = w_k$. Hence, $v'_k = v_k$. Assume now that the assertion is true for all j, $0 \leqslant j \leqslant i$, for some i, $0 \leqslant i \leqslant n_1 - 1$. One proves it for $i+1$. If $M'_{i+1} = \emptyset$ (this means that at the $(i+1)$th step only communications toward the assistant grammars are performed), then the assertion is trivially true. Assume that $M'_{i+1} \neq \emptyset$ and take $k \in M'_{i+1}$. Then for each query symbol $Q^{(s)}_{t2}$ which appears in w'_k, $1 \leqslant t \leqslant n$, $2 \leqslant s \leqslant 2n-2$, s an even number, the following two statements are true: (1) Q_t occurs in w_k; (2) after the ith communication step, in the component $G^{(s)}_{t2}$ there are no more query symbols. This means that all the query symbols in w'_t have been satisfied at a previous step, denote it by j, and the obtained string v'_t has already arrived in a component $G^{(s')}_{t2}$, $s' \geqslant s$. Hence in the grammar $G^{(s)}_{t2}$ either the string v'_t or the axiom S is present. At the $(i+1)$-th communication step the current string in $G^{(s)}_{t2}$ replaces any occurrence of the symbol $Q^{(s)}_{t2}$ in w'_k. As $v'_k \in (N \cup T)^*$ and S is not in $N \cup T$, it follows that the communicated string is v'_t. But $t \in M'_j$, $j \leqslant i$. According to the inductive hypothesis $v'_t = v_t$, hence the replacement of $Q^{(s)}_{t2}$ by v_t in the sentential form w'_k corresponds to the replacement of Q_t by v_t in the sentential form w_k, in the system γ. The above reasoning is valid for any query symbol $Q^{(s)}_{t2}$ in w'_k, hence $w'_k = v_k$. Thus the proof by induction is over. Consequently, Assertion 1 is true.

Consequently, if all the strings obtained after the communication sequence (7) $\Rightarrow^*$ (8) are words over the alphabet of the grammar system γ (words without query symbols), then these communications either represent a simple transfer of the sentential form from G_{k2} to G_{k1}, $1 \leqslant k \leqslant n$, or they correspond to a sequence of communication steps in a terminal derivation in γ, too. A similar conclusion is also true for the sequence of communication steps (4) $\Rightarrow^*$ (5).

All the above considerations lead to the conclusion that, if D' is a terminal derivation in γ', whose configuration do not have in the components G_{kj}, $1 \leqslant k \leqslant n$, $j = 1, 2$, other symbols than the axiom, the query symbols and the symbols in $N \cup T$, then there is a derivation in γ which produces the same string.

Let us now see what happens if in the derivation D', at one moment, the sentential form of a grammar G_{kj}, $1 \leqslant k \leqslant n$, $j = 1, 2$, has a symbol which is not in $N \cup T \cup K' \cup \{S_{kj}\}$. From the above analysis of the work of the system γ', it follows that such a symbol could appear only after a communication, hence it can be only S (the axiom of the assistant grammars). This symbol will not be rewritten in this derivation since the sentential form which circulates between G_{k1} and G_{k2} does not stop in any of the assistant grammars for a rewriting step (and S can be rewritten only in such a grammar). Consequently, the sentential form in which S appears does not participate anymore at the generation of the string $w \in T^*$ produced by D'. This time the components of γ' in whose strings no occurrences of S are present, keep on simulating the behaviour of their correspondents in the system γ. Hence, if in G_{11} the word $w \in T^*$ is produced at one moment, then there is a derivation in γ which produces w, too.

In conclusion, $L(\gamma') \subseteq L(\gamma)$.

($\supseteq$) For the opposite inclusion consider a terminal derivation D in γ. From the form of the rules in γ' it is easy to see that any rewriting step in γ can be simulated in γ'. The following assertion shows that the same is happening with the communication steps, too.

Assertion 2. If $(z_1,\ldots,z_n) \Rightarrow (w_1,\ldots,w_n) \Rightarrow^* (v_1,\ldots,v_n)$ is a derivation in γ, which starts with a rewriting step and continues with communication steps until all the query symbols are satisfied, then there is a derivation in γ' of the form (6) $\Rightarrow$ (7) $\Rightarrow^*$ (8), for which $v'_k = v_k$, $1 \leqslant k \leqslant n$, and a derivation of the form (3) $\Rightarrow$ (4) $\Rightarrow^*$ (5), such that x_k from the configuration (3) is now z_k, the string y'_k from (4) is here w'_k (one has defined w'_k starting from w_k) and z'_k from (5) is now v_k (here z'_k has no connection with z_k), $1 \leqslant k \leqslant n$.

Proof. We shall prove the existence of a derivation in γ' of the first form (for the second case the proof is similar). For the sequence of communication steps in γ, $(w_1,\ldots,w_n) \Rightarrow^* (v_1,\ldots,v_n)$ we make the following notations: M_0 is the set of all indices k, $1 \leqslant k \leqslant n$, for which the string w_k may be communicated at the first step (hence, $|w_k|_K = 0$); n_2 is the number of the communication steps (it follows that $n_2 \leqslant n-1$) and for each i, $1 \leqslant i \leqslant n_2$, M_i is the set of all indices k, $1 \leqslant k \leqslant n$, for which all the query symbols occurring in w_k are satisfied at the ith step.

Assume that r_k is the rule applied at the rewriting step $z_k \Rightarrow w_k$ in the grammar G_k, $1 \leqslant k \leqslant n$. For each k, $1 \leqslant k \leqslant n$, consider the rule $r'_k \in P_{k2}$, $r'_k = A \to u_1 Q_{i_1 2}^{(s_1)} \cdots u_m Q_{i_m 2}^{(s_m)} u_{m+1}$, where $r_k = A \to u_1 Q_{i_1} \cdots u_m Q_{i_m} u_{m+1}$, $m \geqslant 0$, $1 \leqslant i_1,\ldots,i_m \leqslant n$, $u_1,\ldots,u_{m+1} \in (N \cup T)^*$, and for each q, $1 \leqslant q \leqslant m$, $s_q = 2(i-j)$, i,j, being such that $k \in M_i$, $i_q \in M_j$ (clearly, $j < i$, hence $s_q \geqslant 2$; also $s_q \leqslant 2n-2$ as $i \leqslant n_2 \leqslant n-1$). Consider the rewriting step in γ' of the form (6) $\Rightarrow$ (7) performed by applying the rule r'_k in any component G_{k2}, $1 \leqslant k \leqslant n$. It follows that, for each k, $1 \leqslant k \leqslant n$, w'_k is obtained from w_k by replacing any occurrence (if there is any) of a query symbol Q_t by $Q_{t2}^{(s)}$, where $s = 2(i-j)$, $k \in M_i$, $t \in M_j$. After obtaining the configuration (7) in γ', a sequence of communications follows and the final configuration is of the form (8). We use the notations in the proof of Assertion 1 for the sequence of communication steps (7) $\Rightarrow^*$ (8) $(n_1, M'_0,\ldots,M'_{n_1})$. We shall prove by induction on i, $0 \leqslant i \leqslant n_2$, that $M_i \subseteq M'_{2i}$ and $v'_k = v_k$, $k \in M_i$. Let $i = 0$. It is clear that $M_0 = M'_0$ and $v'_k = w'_k = w_k = v_k$, $k \in M_0$. Assume that the assertion is true for all j, $0 \leqslant j \leqslant i$, for an arbitrary i, $0 \leqslant i < n_2$. We shall prove it for $i+1$. Let $k \in M_{i+1}$. If the query symbol Q_t appears in w_k then: (1) all the query symbols in w_t have already been satisfied (hence, there is j, $0 \leqslant j \leqslant i$, such that $t \in M_j$); (2) the symbol $Q_{t2}^{(s)}$ occurs in w'_k, where $s = 2(i+1-j)$. From (1) it follows that at the beginning of the $(i+1)$-th communication step in γ, the string in the component G_t is v_t. According to the inductive hypothesis $M_j \subseteq M'_{2j}$ holds, hence, $t \in M'_{2j}$ and $v'_t = v_t$. Consequently, all the query symbols in w'_t have been satisfied at the $2j$th communication step producing thus the string v_t. This one, after the $(2i+1)$-th communication step, is present in the component $G_{k2}^{(2i+2-2j)} = G_{k2}^{(s)}$. It follows that the

symbol $Q_{t2}^{(s)}$ (which occurs in w'_k) can be satisfied only starting with the $(2i+2)$-th communication step and, if it is satisfied at this step, then it is replaced by v_t (Q_t in w_t is replaced by v_t, too). Since for any occurrence of a query symbol $Q_{t'2}^{(s')}$ in w'_k there is an occurrence of $Q_{t'}$ in w_k, it follows, according to the above reasoning, that all the query symbols of w'_k will be satisfied at the $(2i+2)$-th communication step and the obtained string v'_k is equal to v_k. Now the proof by induction is over. As $M_0 \cup M_1 \cup \ldots M_{n2} = \{1,\ldots,n\}$, it follows that for any $k \in \{1,\ldots,n\}$ there is an i such that $k \in M_i$. By the statement proved above $v'_k = v_k$ holds for $k \in M_i$. In conclusion, Assertion 2 is true.

Following closely the work of the system γ' (explained in the first part of the proof) and applying Assertion 2, one can easily see that for each terminal derivation in γ there is a derivation in γ' generating the same word. Consequently, $L(\gamma) \subseteq L(\gamma')$.

For the right-linear case the same construction is valid with a slight modification. This consists of the replacement of the component G_a by $n(2n-1)$ components $G_{k4}^{(s)} = (N' \cup K', T, P_{k4}^{(s)}, S)$, $1 \leqslant k \leqslant n$, $1 \leqslant s \leqslant 2n-2$, where $P_{k4}^{(1)} = \{S \to Q_{k2}^{(1)}, Z' \to Z'\}$, $P_{k4}^{(s)} = \{S \to Q_{k2}^{(s)}, Z \to Z\}$, $1 \leqslant k \leqslant n$, $2 \leqslant s \leqslant 2n-2$. These grammars have altogether the same role as G_a. The new grammar system has only right-linear rules and generates the same language as the initial one.

References

[1] L. Cai, The computational complexity of PCGS with regular components, *Proc. Developments in Language Theory Conf.*, Magdeburg (1995).

[2] Csuhaj-Varju, J. Dassow, J. Kelemen and Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation* (Gordon and Breach, London, 1994).

[3] S. Dumitrescu and Gh. Păun, On the power of parallel communicating grammar systems with right-linear components, submitted.

[4] R. Freund, Gh. Păun, C.M. Procopiuc and O. Procopiuc, Parallel communicating grammar systems with context-sensitive components, in: Gh. Păun, ed., *Artificial Life. Grammatical Models* (The Black Sea Univ. Press, Bucharest, 1995) 166–174.

[5] J. Hromkovic, Descriptional and computational complexity measures for distributive generation of languages, in: *Proc. of Developments in Language Theory Conf.*, Magdeburg (1995).

[6] J. Hromkovic, J. Kari, L. Kari and D. Pradubska, Two lower bounds on distributive generation of languages, in: *Proc. MFCS 94, LNCS 841* (Springer, Berlin, 1994) 423–432.

[7] V. Mihalache, On parallel communicating grammar systems with context-free components, in: Gh. Păun, ed., *Mathematical Linguistics and Related Topics* (The Publ. House of the Romanian Academy, Bucharest, 1995) 258–270.

[8] D. Pardubska, Communication complexity hierarchies for distributive generation of languages, *Fundamenta Informaticae*, to appear.

[9] Gh. Păun and L. Sântean, Parallel communicating grammar systems: the regular case, *Ann. Univ. Buc., Ser. Matem.-Inform.* **38** (1989) 55–63.

[10] O. Procopiuc, C.M. Ionescu and F.L. Ţiplea, Parallel communicating grammar systems: the context-sensitive case, *Internat. J. Comput. Math.* **49** (1993) 145–156.

[11] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).

[12] G. Vaszil, The simulation of a non-returning parallel communicating grammar system with a returning system in case of linear component grammars, submitted.